Attracting Contributions to your GitHub Project

Javier Luis Cánovas Izquierdo, Valerio Cosentino and Jordi Cabot
AtlanMod team (Inria, Mines Nantes, LINA), Nantes, France
{javier.canovas,valerio.cosentino,jordi.cabot}@inria.fr

ABSTRACT

Most Open Source Software projects can only progress thanks to developers willing to voluntarily contribute. Therefore, their vitality and success largely depend on their ability to attract developers. Code hosting platforms like GitHub aim at making software development more collaborative and attractive for contributors by providing facilities such as issue-tracking, code review or team management on top of a Git repository following a pull-based model to handle external contributions. We study whether the use of these facilities actually help to get more contributions based on a quantitative analysis over a dataset composed by all the GitHub projects created in the last two years. We discovered that most projects actually ignore them and that, those that don't, do not advance faster either. A manual analysis of the most successful projects suggests that other factors like clear description of the contribution and governance rules for the project have a greater impact.

Categories and Subject Descriptors

H.4.0 [**Information Systems Applications**]: General; D.2.8 [**Software Engineering**]: Metrics; K.6.1 [**Management of Computing and Information Systems**]: Project and People Management

General Terms

Management, Measurement

Keywords

Open Source Software, GitHub, Contribution analysis

1. INTRODUCTION

The vitality and success of Open Source Software (OSS) projects depend on their ability to attract, absorb and retain new developers [1]. In the last years, websites such as GitHub have positively contributed to the rise of OSS by providing code hosting platforms aiming at promoting OSS projects and facilitating the collaboration thanks to facilities such as team management support, issue-tracking and a pull-based model implementation.

GitHub enables a distributed development model based on Git (though with some extensions). In GitHub there are two main development strategies aimed at (1) the project team members and (2) external developers. Team members have direct access to the source code, which they modify by means of *push*es. External developers follow a pull-based model, where any developer can work isolately with clones (facilitated by means of *fork*s in GitHub) of the original source code. Later, developers can then send back their changes and request those changes to be integrated in the project codebase. This is what is called to send a *pull request*s. Finally, pull requests are evaluated by project team members, who can either approve the pull request and incoporate the changes, or reject it and propose improvements which can be addressed by the proponent. Beyond the project creator, other developers can be promoted to the status of official project collaborators and get most of the same rights project owners have, so that they can help not only on the development (by means of pushes, as said above) but also with management tasks (e.g., answering issues or providing support to other developers). Issue-tracking support helps both external developers and team members to request new features and report bugs and therefore fosters the participation in the development process.

There is still a very limited understanding about what makes some OSS projects advance faster than others. In this paper we are interested in studying whether the collaboration facilities provided by code hosting platforms like GitHub influence in the advancement of the project. We conducted a quantitative analysis considering all the GitHub projects created in the last two years. Several works have performed qualitative analysis of GitHub samples ([7, 5, 10] among others), however, to the best of our knowledge, ours is the largest quantitative study on GitHub. We analyzed each project to study whether these facilities actually help to make the project advance. In the context of this work, the project advance is measured in terms of development advance (i.e., commits). Finally, we complemented our study with a manual analysis over a sample of successful projects to reason about our results.

The paper is organized as follows. Section 2 describes the methodology followed throughout the paper. Section 3 presents the evaluation while Section 4 describes the threats to validity. Section 5 discusses the results and the future work. Section 6 presents the related work and Section 7

ASE conference '14 Sweden

ends the paper.

2. METHODOLOGY

In this section we describe the methodology we followed to perform our study, based on the analysis of all the projects created in GitHub since January 2012. We first describe the set of attributes we want to analyze for each GitHub project. We then describe how we built a dataset containing the attribute values for each project. Finally, we present our hypotheses and research questions, which will be evaluated in Section 3.

2.1 GitHub Project Attributes

For each project in our dataset we are interested in getting insights regarding the following characteristics:

1. **General information**. We consider basic project information such as whether the project is a fork of another (`forked` attribute) and the programming language used (`language` attribute).

2. **Development**. We measure the development status of GitHub projects in terms of commits (`totalCommits` attribute) since its creation. As GitHub projects can receive commits from pushes (i.e., source code contributions coming from team members) and pull requests (i.e., source code contributions coming from accepted pull requests), we distinguish `commitsPush` and `commitsPR` attributes, respectively.

3. **Interest**. Being a social coding site, GitHub projects can also be monitored, tracked and forked by users. We therefore focus on two main facilities provided by GitHub: watchers (`watchers` attribute) and forks (`forks` attribute). The former is the number of people interested in following the evolution of the project; they are notified when the project status changes (e.g., new releases, new issues, etc.). The latter is the number of people that made a fork. Both attributes can provide good insights on the project popularity [3].

4. **Collaborators**. We consider the number of collaborators (`collabs` attribute) who have joined a project to help in its development.

5. **Contributions**. We focus on contributions coming from (1) pull requests (`PRs` attribute) and (2) issues (`issues` attribute). In particular, we are interested in collecting the number of pull requests and issues that have been proposed (i.e., opened) for each project.

2.2 Mining GitHub

The mining process is illustrated in Figure 1 and is composed of three phases: (1) extracting the data (see *Extractor*), (2) aggregating the data to calculate and import the attribute values for each project into a database (see *Aggregator*), and (3) filtering the database to build the subset of projects used for analysis (see *Filter*). Next, we will describe each phase of the process.

Extractor. The data regarding GitHub projects has been obtained from GitHub Archive[1], which has tracked every public event triggered by GitHub since February 2011.

[1] `http://www.githubarchive.org`

Table 1: Events considered in the GitHub Archive extractor.

Event Type	Triggering condition	Attributes Involved
MemberEvent	A user is added as a collaborator to a repository	`collabs`
PushEvent	A user performs a push	`commitsPush`
WatchEvent	A user stars a repository	`watchers`
PullRequestEvent	A pull request is created, closed, reopened or synchronized	`PRs`, `commitsPR`
ForkEvent	A user forks (i.e., clones) a repository	`forks`
IssuesEvent	An issue is created, closed or reopened	`issues`

GitHub events describe individual actions performed on GitHub projects, for instance, the creation of a pull request or a push. Events are represented in JSON format following a similar structure: (1) a set of common elements describing general information of the project (i.e., name, owner, language, whether the project was forked, etc.) and (2) a *payload* object containing the specific information of the event.

There are 22 types of events but we focus on 7 of them[2], from which we can get the data needed to calculate the project attributes described before. The considered event types are presented in Table 1.

Since we want to study the projects created in the last two years, the extractor retrieves all the events from January 1^{st} 2012 until February 25^{th} 2014. As events are stored in GitHub Archive hourly, the process collected all the events triggered per day and stored them in a file (in average, half a millon of events were collected per day). Furthermore, since we want to analyze only those projects for which we have all the events since their creation, events of projects created before January 1^{st} 2012 were removed. In total, the extractor collected the events for 18888 hours (i.e., 787 days), of which 29 hours could not be retrieved (less than 0,15% of the total) due to missing hours in GitHub Archive.

Aggregator. This component aggregates the events extracted in the previous step and calculates the attributes for each project. The implemented aggregator creates a database with a single table including a column for each attribute presented before plus two more columns to store the project name and owner. Attributes regarding the project general information are calculated from the common elements included in the events, while the rest are calculated from specific events (as indicated in the last column of Table 1).

The resulting dataset contains 7,760,221 projects. This dataset was curated to solve two problems, specifically: (1) empty values either in the project name or owner fields, and (2) the use `PublicEvent` event. The former was detected in the JSON events extracted from GitHub Archive and affected 13,138 projects (less than 0.17% of the total number of projects). On the other hand, the use of the `PublicEvent` means that the project was private and later become public, thus entailing the loss of events while it was private. In total, 322,460 projects used the `PublicEvent` event (around 4.15% of the total number of projects). We removed the projects

[2] The complete list of events can be found at `https://developer.github.com/v3/activity/events/types`

Figure 1: Mining process.

showing these two problems. The curated dataset contained 7,365,622 projects.

Filter. This component allows building subsets of the previous dataset in order to perform a more focused analysis. The filter takes as input the dataset from the previous step and creates a new filtered dataset containing only those elements fulfilling a particular condition.

In the context of our study, we built a new filtered dataset including only those projects not being a fork of another and using a programming language. GitHub is used for many other tasks beyond software development (i.e. writing books) and we wanted to focus only on original software development projects. The resulting filtered dataset contained 2,126,093 projects and was the one used in all the other analysis presented in this paper.

2.3 Hypotheses and Research Questions

The goal of our study is to obtain a deeper understanding of which factors help some projects to attract more contributions and thus advance faster, specially focusing on the collaboration facilities provided by GitHub. These facilities are measured by the `collabs`, `issues` and `PRs` attributes (as described above), which allow us to track the use of team management, issue-tracking and pull-based facilities, respectively, in a project. To this end, we are interested in first studying whether such facilities are used and then how they affect the project success. In the context of this work, the project success is measured in terms of advancement in the development (i.e., commits). To complement the analysis we will also evalute whether the interest a project raises influences in its success as well.

As a result, we make the following hypotheses that we will try to (in)validate in this work.

H1 Projects use the collaboration facilities provided by GitHub.

H2 The more the collaboration facilities are used in a project, the more successful the project is.

H3 The more interest a project attracts, the more successful the project is.

We have identified the following research questions which will help us to (in)validate the previous hypotheses:

RQ1 *Are collaboration facilities being used in GitHub?*. To answer this question, we will characterize GitHub projects according to the attributes presented before and specifically study the use of collaboration facilities (i.e., H1).

RQ2 *Is there a relationship between the project success and either the (a) the collaboration facilities (i.e., H2) or (b) the project interest attributes (i.e., H3)?*. To answer this question, we will perform a correlation analysis among the involved attributes.

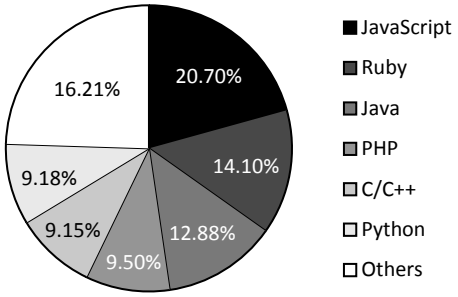

Figure 2: Programming languages used in the dataset.

Table 2: Project attributes results of the GitHub dataset.

Development attributes						
Attribute	**Min.**	**Q1**	**Median**	**Mean**	**Q3**	**Max.**
`totalCommits`	0.00	2.00	7.00	43.00	19.00	5545441.00
`commitsPush`	0.00	2.00	7.00	41.00	19.00	5545441.00
`commitsPR`	0.00	0.00	0.00	1.31	0.00	38242.00
Interest attributes						
Attribute	**Min.**	**Q1**	**Median**	**Mean**	**Q3**	**Max.**
`watchers`	0.00	0.00	0.00	2.26	1.00	14607.00
`forks`	0.00	0.00	0.00	0.68	0.00	2913.00
Collaborators and Contribution attributes						
Attribute	**Min.**	**Q1**	**Median**	**Mean**	**Q3**	**Max.**
`collabs`	0.00	0.00	0.00	0.05	0.00	7.00
`PRs`	0.00	0.00	0.00	0.96	0.00	8337.00
`issues`	0.00	0.00	0.00	0.29	0.00	1540.00

3. EVALUATION

In this section we report the results of our study, which allow us to answer the previous research questions.

3.1 RQ1: GitHub Characterization

In the first research question we investigate the main characteristics of GitHub projects with regards to the project attributes presented in Section 2.1. Figure 2 shows a pie chart including the most used programming languages. As can be seen, the great majority of projects are being developed in JavaScript, Ruby and Java, which represent around 48% of the projects considered in our dataset.

A summary of the results for the other project attribute values (computed directly querying the dataset obtained in the mining process) is shown in Table 2. As can be seen, the results reveal very low values for each considered attribute (see the Q1, Mean and Q3 values). We now discuss in detail each group of attributes.

The results for development attributes such as `totalCommits` are strongly influenced by the fact that a considerable number of projects have a small number of commits. Thus, 1,259,822 (59.26% of the total number of projects) have between 0-10 commits from pushes

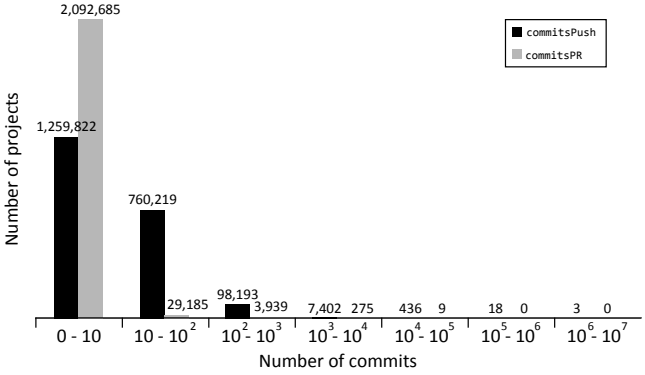

Figure 3: Comparison between number of projects and number of commits coming from pull requests (`commitsPR`) and pushes (`commitsPush`).

Table 3: Project analysis according to the usage of GitHub collaboration facilities.

			Projects		Commits		
Collabs	PRs	Issues	#	%	Mean	St. Dev.	Median
No	No	No	1,737,471	81.72%	31.00	4,463.34	5
No	No	Yes	57,575	2.71%	57.06	784.37	16
No	Yes	No	124,170	5.84%	71.64	379.73	17
No	Yes	Yes	98,695	4.64%	202.20	2,933.21	39
Yes	No	No	82,246	3.87%	34.60	260.52	10
Yes	No	Yes	5,499	0.26%	82.47	254.79	29
Yes	Yes	No	10,090	0.47%	74.50	428.66	29
Yes	Yes	Yes	10,347	0.49%	226.20	1,282.00	62

(`commitsPush`) and 2,092,685 (98.47% of the total number of projects) have only between 0-10 commits from pull requests (`commitsPR`). Figure 3 illustrates this situation by showing the number of projects (vertical axis) per group of commits (horizontal axis). Regarding the interest attributes, 1,433,042 projects (67.40% of the total number of projects) have 0 `watchers` and 1,614,556 projects (75.94% of the total number of projects) have never been forked. These results suggest that the use of GitHub is far from what it would be expected as a social coding site.

The results for collaborator and contribution attributes also reveal a very poor usage. Thus, 2,017,911 projects (94.91% of the total number of projects) do not use the collaborator figure; 1,953,977 projects (91.90% of the total number of projects) have never received a pull request; and 1,949,644 projects (91.70% of the total number of projects) have never received an issue.

To better characterize the dataset, we have grouped the projects according to whether they use or not each collaboration facility. Table 3 shows the results obtained. As can be seen, the biggest group of projects corresponds to those ones not using any collaboration facility (i.e., see first row, 1,737,471 projects, the 81.72% of the total number of projects). The table also includes the average number of commits of the projects included in each category. As can be seen, the results for those projects using any of the collaboration facilities are better than for those not using them. This result suggests that the use of collaboration facilities may increase the development attribute values (i.e., commits). However, a more detailed study focusing on the real correlation of these attributes in provided in the next section.

In order to gain some insights that could help us to better interpret these results, we started a discussion thread in Reddit[3]. Most of the participants acknowledged that they use GitHub for backup or curriculum vitae purposes, so even if the projects were publicly available (to avoid paying the fees required to have them with restricted access), they never really intended to look for contributions.

> A project characterization analysis in GitHub reveals that collaboration facilities are very scarcely used. This finding invalidates H1, our first hypothesis presented before. Furthermore, the great majority of projects show a low activity (i.e., `totalCommits`, `commitsPush` and `commitsPR`) and attract low interest (i.e., `forks` and `watchers`).

3.2 RQ2: Correlation Analysis

For this research question, our objective is to study whether there is a relationship between the involved attributes (`collabs`, `PRs`, `issues`, `watchers`, `forks`, `totalCommits`, `commitsPush`, `commitsPR`). To this end, we resort in the Spearman's rho (ρ) correlation coefficient to confirm the existence of a correlation among the considered attributes. This coefficient is used in statistics as a non-parametric measure of statistical dependence between two variables. The values of ρ are in the range $[-1, +1]$, where a perfect correlation is represented either by a -1 or a $+1$, meaning that the variables are perfectly monotonically related (either increasing or decreasing relationship, respectively). Thus, the closer to 0 the ρ is, the more independent the variables are.

Table 4 shows the ρ values for each combination of the considered attributes. In particular, the first three rows and columns show the correlation values for the collaboration facilities and the development attributes. As can be seen, the low ρ values for the `collab` attribute denotes that it is not correlated with any other attribute. There is a low correlation between `issues` and `commitsPR`, which suggests that issue requests may end up with changes in the source code. Finally, `PRs` and `commitsPR` attributes are strongly correlated ($\rho = 0.88$), which may be obvious if we consider that accepting a pull request implies the incorporation of the set of commits part of that request in the source code. However, as `PRs` indicates the number of pull request opened, this result also suggests that a significant number of open pull requests are finally accepted. The last two rows of Table 4 include the correlation values for the project interest and development attributes. As can be seen, only the `forks` and `commitsPR` attributes are correlated, with a low value though ($\rho = 0.36$).

Additionally, Table 4 also shows the correlation values among the collaboration facilities and interest attributes, which are considered by some authors (i.e., [4, 3]) to measure the success of a project. As can be seen, only the value between `PRs` and `forks` attributes reveals a moderate correlation, which may suggest a proper use of the pull-based development model in GitHub (i.e., forking a project could end up sending a pull request). Finally, Table 4 also includes the correlation value between `watchers` and `forks` attributes, which shows a moderate relationship between them.

[3] `http://goo.gl/sx0Phz`

Table 4: Correlation analysis between the considered attributes.

	Development attributes			Interest attributes	
	totalCommits	commitsPush	commitsPR	watchers	forks
collabs	0.09	0.09	0.06	0.02	0.03
PRs	0.27	0.25	**0.88**	0.26	**0.40**
issues	0.25	0.25	**0.34**	0.28	0.24
watchers	0.11	0.10	0.24	1.00	**0.57**
forks	0.08	0.07	**0.36**	**0.57**	1.00

It is important to note that during our study we also calculated the correlation values among all these attributes when grouping the projects according several dimensions, specially based on their size and the language used. None of those groupings revealed different results from those shown above.

A correlation analysis reveals that among the attributes regarding the collaboration facilities, only `PRs` is highly correlated with `commitsPR` while `issues` is low correlated. H2 is therefore disputed. Regarding the interest attribute, there is a moderate correlation between `PRs` and `forks` attributes. This result disputes H3.

4. THREATS TO VALIDITY

In this section we describe the threats to validity we have identified in our study.

External Validity. Our study considers a large dataset of GitHub projects, however, it may not represent the universe of all real-world projects. In particular, as GitHub allows users to create open source repositories without any expense, our dataset might include mock or personal projects that are not focused on attracting contributions and they have been open sourced only to avoid paying membership fees to keep them private. Further comparative analysis should be conducted against other social hosting coding sites (e.g., BitBucket) allowing free private repositories to contrast our results.

Internal Validity. Our study only considers GitHub data and therefore does not take into account external tools used by some GitHub projects (e.g. to manage the team and issues; for instance people attaching patches to an external Bugzilla bug tracking tool, later manually merged into the project by the project owner) that can lead to bias our study (i.e. in the previous example, that patch would not count as a pull request). Finally, using the `language` attribute to filter out non-software projects may result in the elimination of relevant projects since some software projects do not set the programming language used.

5. DISCUSSION AND FUTURE WORK

According to our results, there is little evidence that the use of GitHub collaboration facilities results in more contributions to the projects. Thus, we extended our study to consider other reasons that could explain why some projects seem better at that than others.

For this, we conducted a manual inspection of the 50 most successful GitHub projects in our dataset (success measured in terms of the number of commits of the project coming from pull requests, i.e., from external contributors). We noticed that 92% of them (i.e., 46 projects) included a description file (i.e. *readme*), with, often, a link to complementary information in wikis (46%) and/or external websites (50%). A further manual inspection of these three kinds of project information sources revealed that they were not purely "decorative" but that instead included precise information on the process to follow for all those willing to contribute to the project (e.g., how to submit a pull request, the decision process followed to accept a pull request or an issue, etc.).

Based on this, it seems reasonable to think that a clear description of the contribution process is a significant factor to attract new contributions. That is, it is not just a matter of whether projects use a certain feature but a matter of how they use it.

Unfortnately, existing GitHub APIs and services do not provide direct support to automatically check our hypothesis on the whole population of GitHub projects. Thus, as early validation, we took two random samples *noPR* and *withPR*, each of them composed by 50 original projects (public and at least one year old) including a project description file. The sample *noPR* contained projects with no commits from pull requests, while the sample *withPR* contained projects with at least one commit from a pull request. We manually analyzed the resources of the projects considered in each sample. Regarding the *noPR* sample, in most of the cases the project resources did not contain any valuable information (only the 16% of the project provided indications to contribute). On the other hand, in the *withPR* sample we noticed a clear improvement in the quality of the project resources and indications (around 60% of the projects included high-quality indications about how to get involved).

We belive this preliminary validation confirms that this is at least a research direction worth exploring further and that opens the door to new interesting research questions. In particular, we are interested in first confirming this relationship between clear description of the contribution process and its impact on the number of contributions received. To this aim, we plan to extend our analysis of GitHub projects with a semi-automatically parsing of project description files (and external websites linked from there). If this is confirmed, we would like to explore by means of surveys and interviews with developers actively contributing to open source projects, whether beyond the existance of a clear contribution path they favour projects that follow a specific collaboration model (i.e. are they more inclined to collaborate in projects where the decision of accepting a pull request is openly discussed, instead of just being the sole decision of the project owner, or not?) This could help project owners to decide whether to have a more transparent governance process in order to advance faster in the project development.

6. RELATED WORK

GitHub data has been the target of a number of research papers. Works such as [6] or GitHub Archive aim to simplify the mining of GitHub data. Recommendations on how to do it are provided in [2]; we have taken them into consideration when performing our analysis.

The pull-based development model in GitHub has been the focus of [7]. Authors concluded that the pull-based model offers increased opportunities for community engage-

ment and decreased time to incorporate contributions. Unlike our work, they did not aim at analyzing how this translated into contributions to the project advancement.

Other works studied attributes that may potentially affect the project advancement, measured in terms of watchers and forks. Thus, [4] and [3] works study whether the use of a particular programming language or issue-tracking systems, respectively, may have an impact in the project. Both conclude that such impact is disputed.

Other works analyze GitHub projects from a social point of view. In [10] authors study the social structure (i.e., social network among developers) of a set of projects and report on the impact in the project transparency (i.e., how clear is for the developer community the development process). The work presented in [5] analyzes the implications of a herding behaviour in GitHub projects. These works can be helpful in the future to identify other possible external attributes influencing the advancement of GitHub projects.

Outside the context of GitHub, some works have analyzed the impact of using issue-tracking systems in OSS projects. In [9] authors study the relationship between improvements, new features and defects recorded in the tracker. The work presented in [8] investigates the use of these systems in open and close projects and reports on the usefulness of making public issue-tracking systems for closed projects.

7. CONCLUSION

We have presented the results of a exploratory study aimed at determining whether the collaboration and social features provided by git-based code hosting environments like GitHub have a real impact on the advancement (as measured by the number of commits performed) of projects hosted therein. The data and processing scripts used in the study are availabe online [4].

Our findings show that in fact most projects ignore these features and just use GitHub as a free backup option. For the few that do try to use them, we found little evidence that their usage positively impacts on the project evolution. A detailed analysis of the most successful projects (successful defined in terms of their ability to attract external contributors) seems to reveal that a key characteristic they all have in common is a clear description of the expected contribution process to the project (sometimes using tools external to GitHub itself), not only covering guidelines for the submission phase, but also including information on how that submission will be later evaluated and processed (accepted, integrated in the next release, etc.) by the project owners. Further work will be continue exploring this hypothesis, as explained in Section 5.

8. REFERENCES

[1] C. Bird, A. Gourley, P. Devanbu, U. C. Davis, A. Swaminathan, and G. Hsu. Open Borders ? Immigration in Open Source Projects. In *MSR conf.*, 2007.

[2] C. Bird, P. Rigby, and E. Barr. The promises and perils of mining git. In *MSR conf.*, pages 1–10, 2009.

[3] T. F. Bissyande, D. Lo, L. Jiang, L. Reveillere, J. Klein, and Y. L. Traon. Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub. *ISSRE symp.*, pages 188–197, 2013.

[4] T. F. Bissyande, F. Thung, D. Lo, L. Jiang, and L. Reveillere. Popularity, Interoperability, and Impact of Programming Languages in 100,000 Open Source Projects. In *COMPSAC conf.*, pages 303–312, 2013.

[5] J. Choi, J. Moon, J. Hahn, and J. Kim. Herding in open source software development: an exploratory study. In *CSCW conf.*, pages 129–133, 2013.

[6] G. Gousios. The ghtorrent dataset and tool suite. In *MSR'13*, pages 233–236, 2013.

[7] G. Gousios, M. Pinzger, and A. V. Deursen. An Exploratory Study of the Pull-based Software Development Model. In *ICSE conf.*, page to appear, 2014.

[8] L. Grammel and H. Schackmann. Attracting the community's many eyes: an exploration of user involvement in issue tracking. *HAOSE Conf*, 2010.

[9] D. Posnett, A. Hindle, and P. Devanbu. Got Issues? Do New Features and Code Improvements Affect Defects? In *CSMR conf*, pages 211–215, 2011.

[10] F. Thung, T. F. Bissyande, D. Lo, and L. Jiang. Network Structure of Social Coding in GitHub. In *CSMR conf.*, pages 323–326, 2013.

[4] `https://github.com/atlanmod/githubContributionAnalysis`